Memory Analysis and CPU-time Profiling in RMG-Java

Kehang Han
Jan. 22, 2014

Outline

Memory Management in Java

Demo of Memory Analysis

CPU-time Profiling

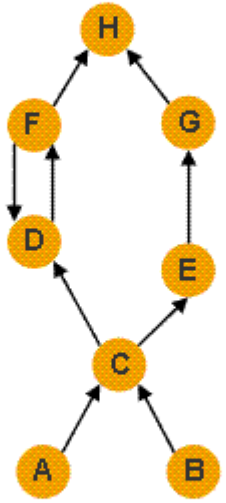
Possible Approaches

Memory Management in Java

- Programming languages like C/C++
    - Manually allocate/de-allocate memory
- Java
    - Automatically de-allocate
    - Garbage collector

Basic concepts for Garbage Collection

- Heap dump



- Shallow heap

Memory consumed by one object itself

- G.C. root

Any variables your program can access directly

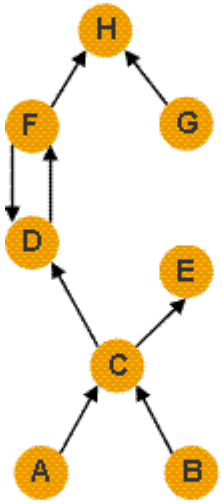
- Local variables
- Class static variables

Basic concepts for Garbage Collection

- Live objects

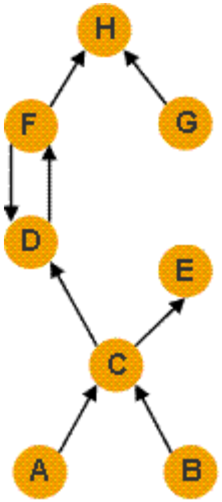
Can be reached from G.C. Root

- Retained set & heap

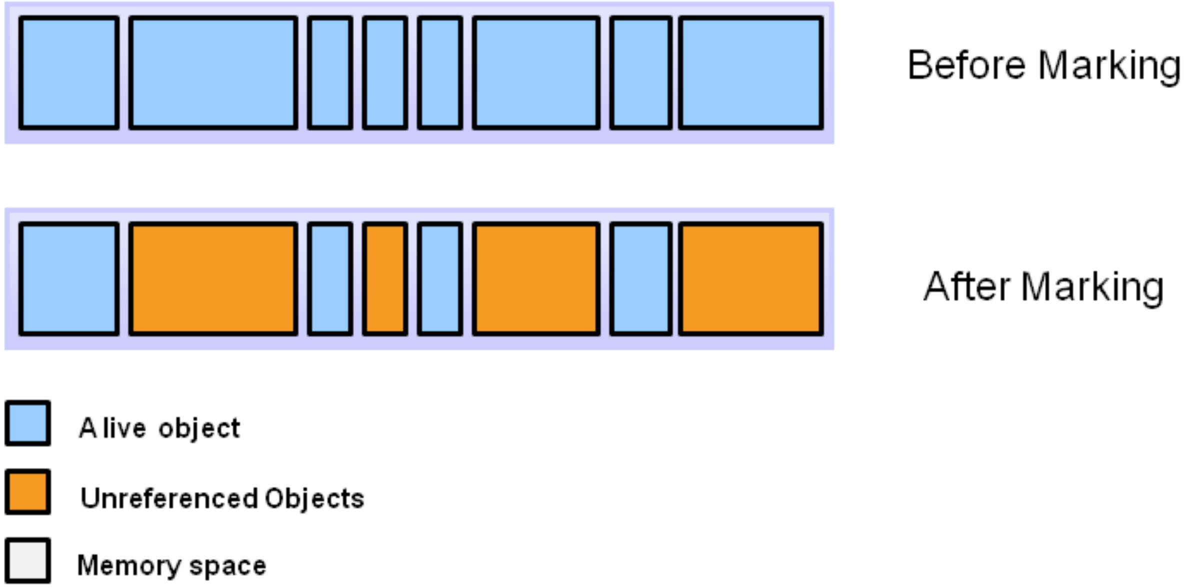


- Dominator

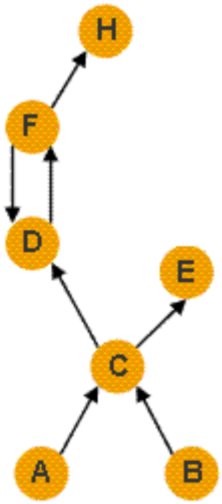
Mark and Sweep Garbage Collection



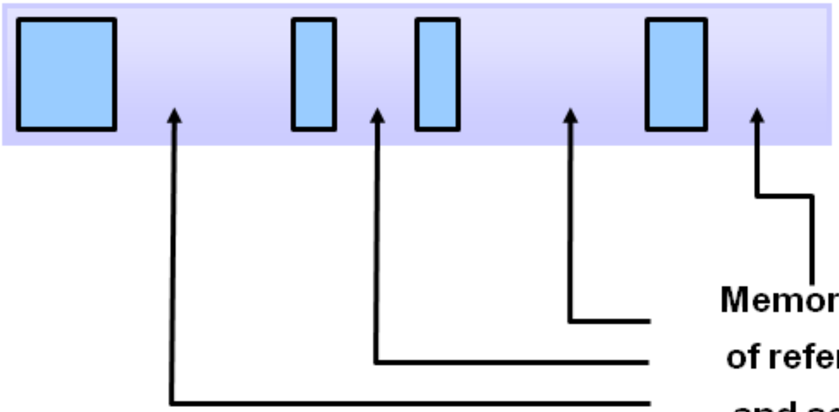
Marking



Mark and Sweep Garbage Collection



Normal Deletion

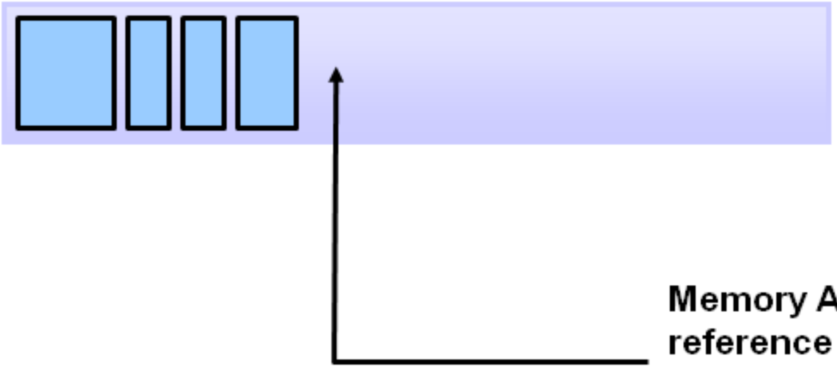


After normal deletion

Memory Allocator holds a list of references to free spaces, and searches for free space whenever an allocation is required

Mark and Sweep Garbage Collection

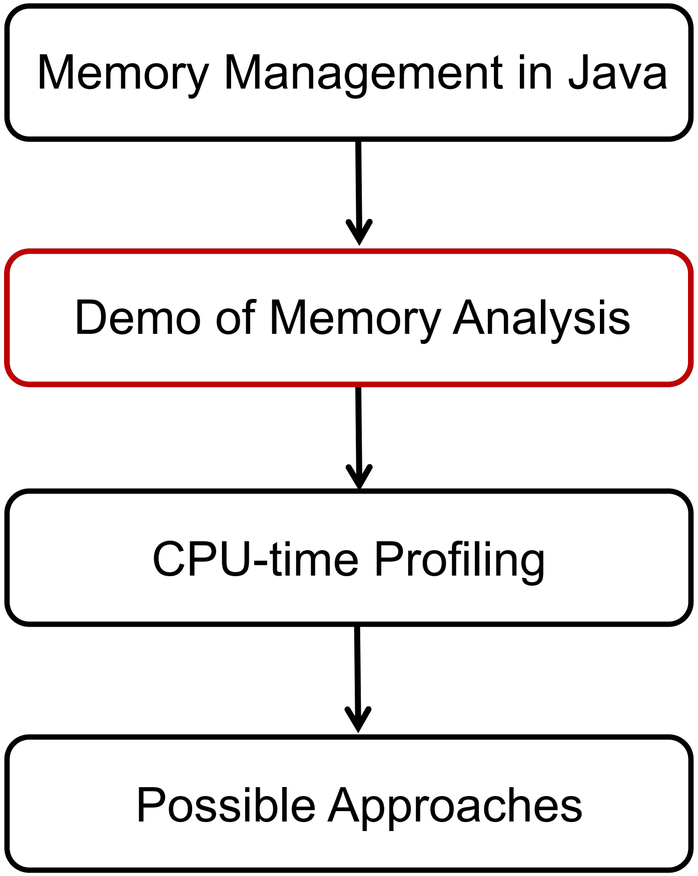
Deletion with Compacting



After normal
Deletion with
compacting

Memory Allocator holds the reference to the beginning of free space, and allocated memory sequentially then on.

Outline



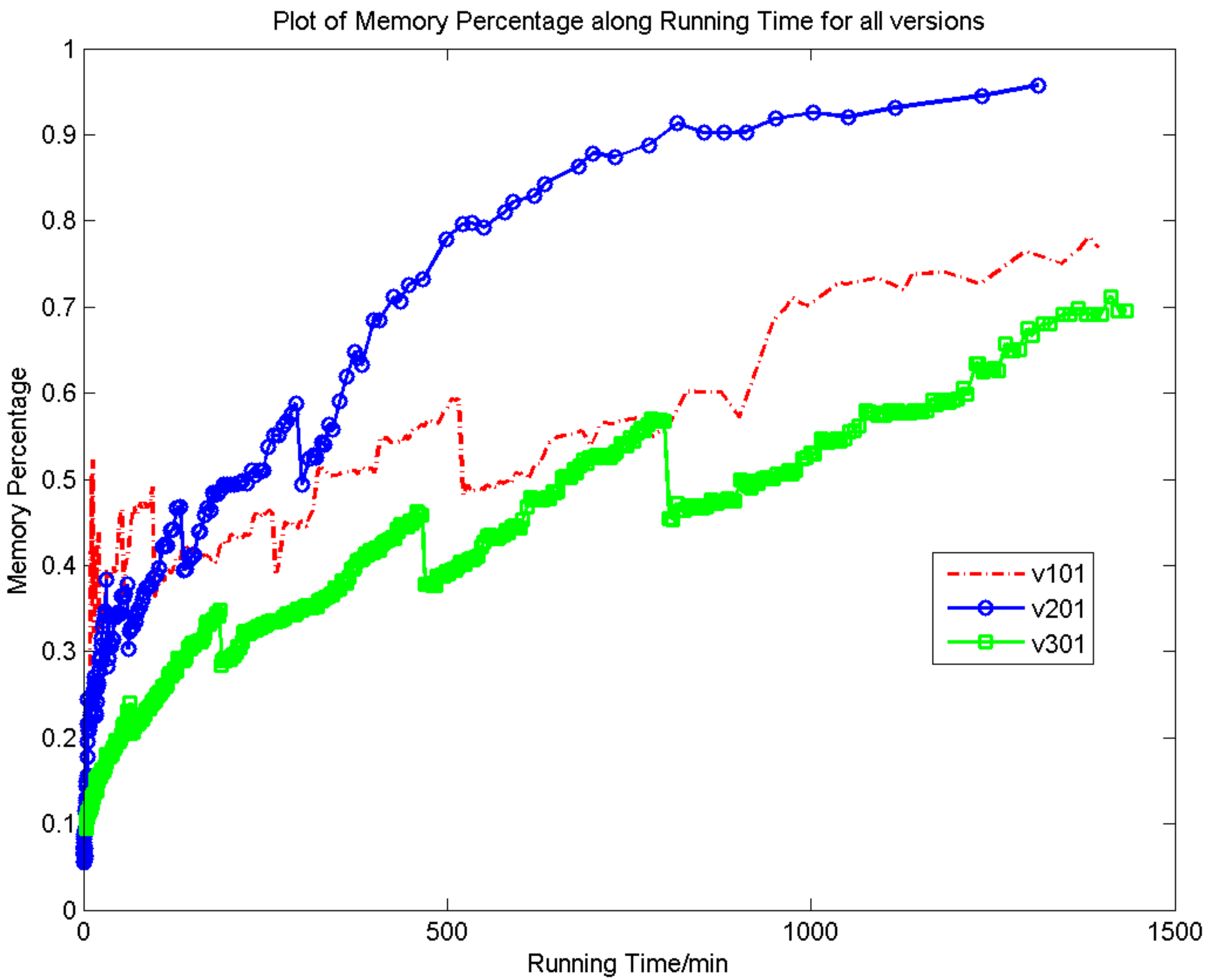
Memory Management in Java

Demo of Memory Analysis

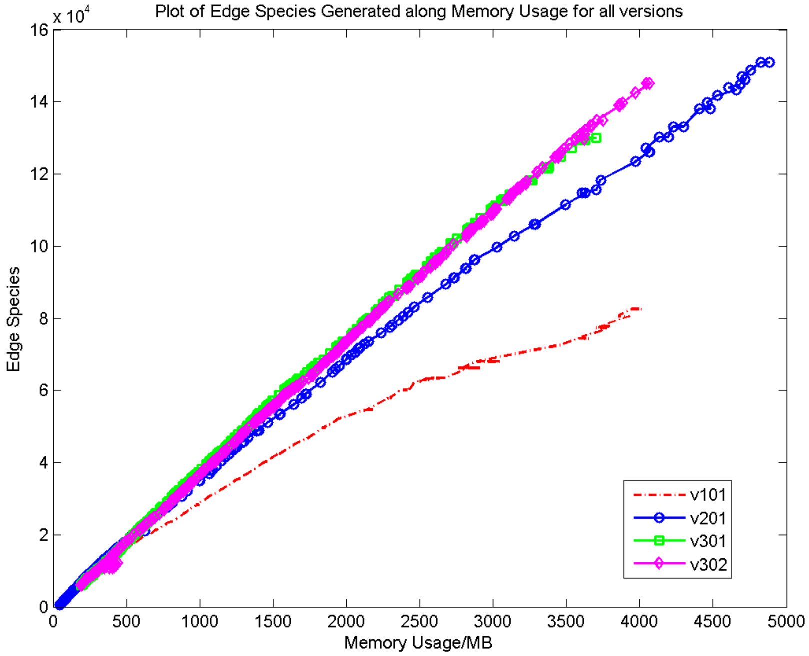
CPU-time Profiling

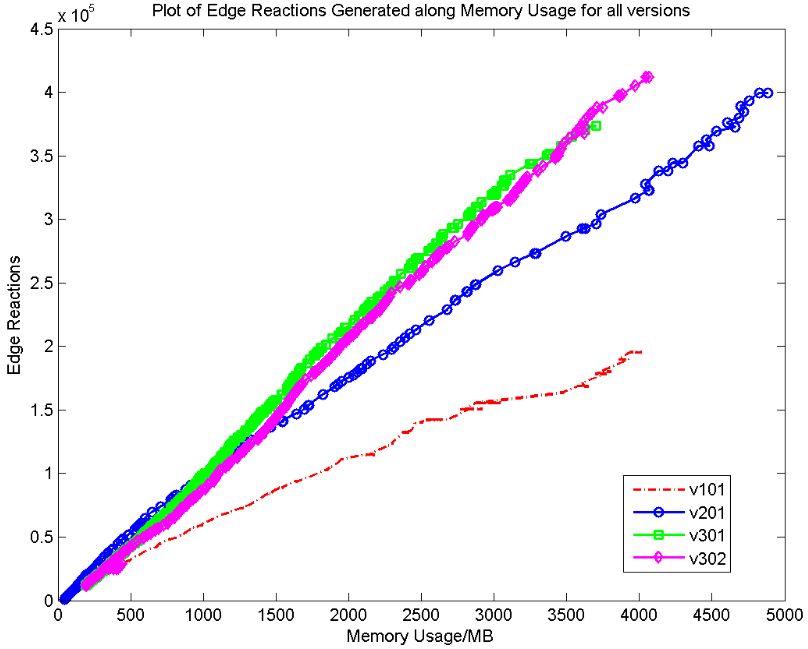
Possible Approaches

RAM limitation



RAM limitation



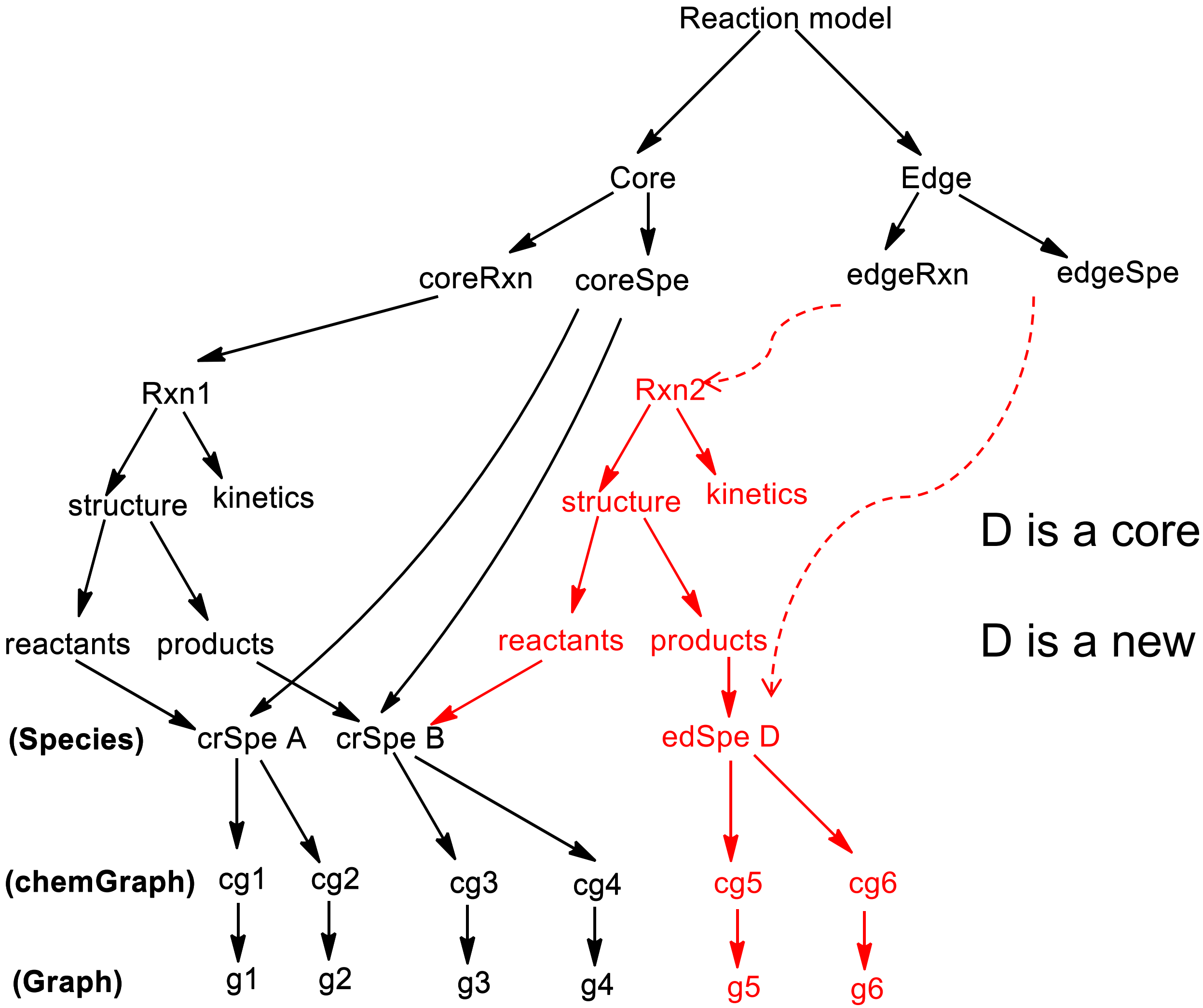


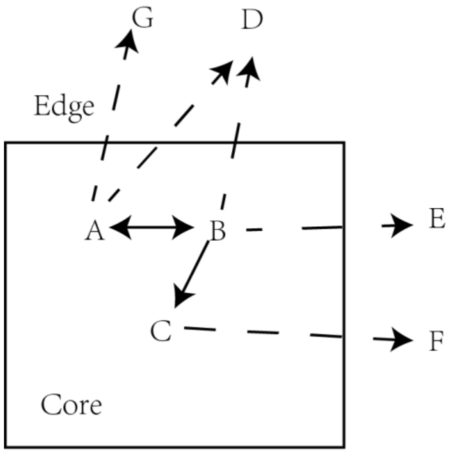
Demo of Memory Analysis

- How to get a heap dump
  - Console: `jmap -dump:format=b,file=<filename.hprof> <pid>`
  - .sh file: `-XX:+HeapDumpOnOutOfMemoryError`

- How to use Eclipse Memory Analyzer
  - Histogram
  - Outgoing & incoming
  - Dominator tree & immediate dominator
  - Retained set

Object Graph in RMG-Java

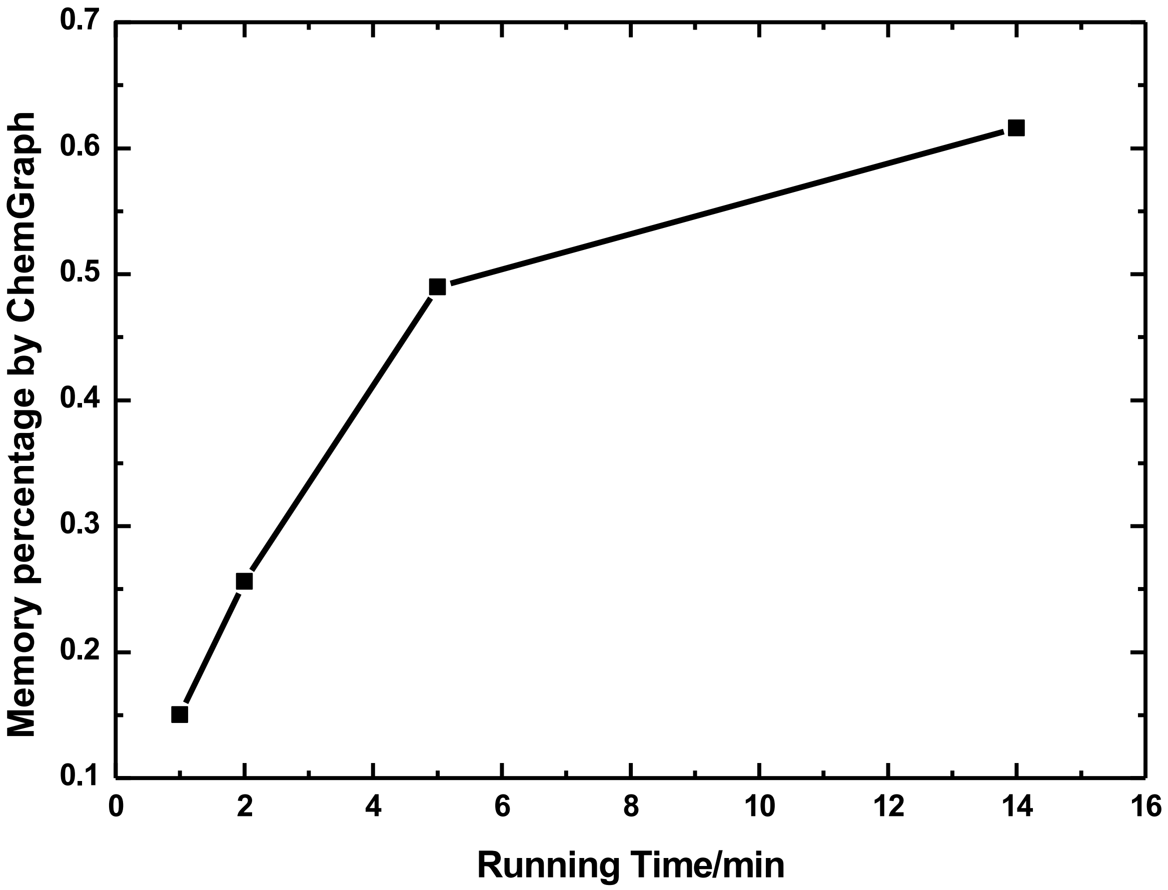

Reaction model
Core
Edge
coreRxn
coreSpe
edgeRxn
edgeSpe
Rxn1
Rxn2
structure
kinetics
structure
kinetics
reactants
products
reactants
products
(Species) crSpe A crSpe B edSpe D
(chemGraph) cg1 cg2 cg3 cg4 cg5 cg6
(Graph) g1 g2 g3 g4 g5 g6



D is a core species? ✗

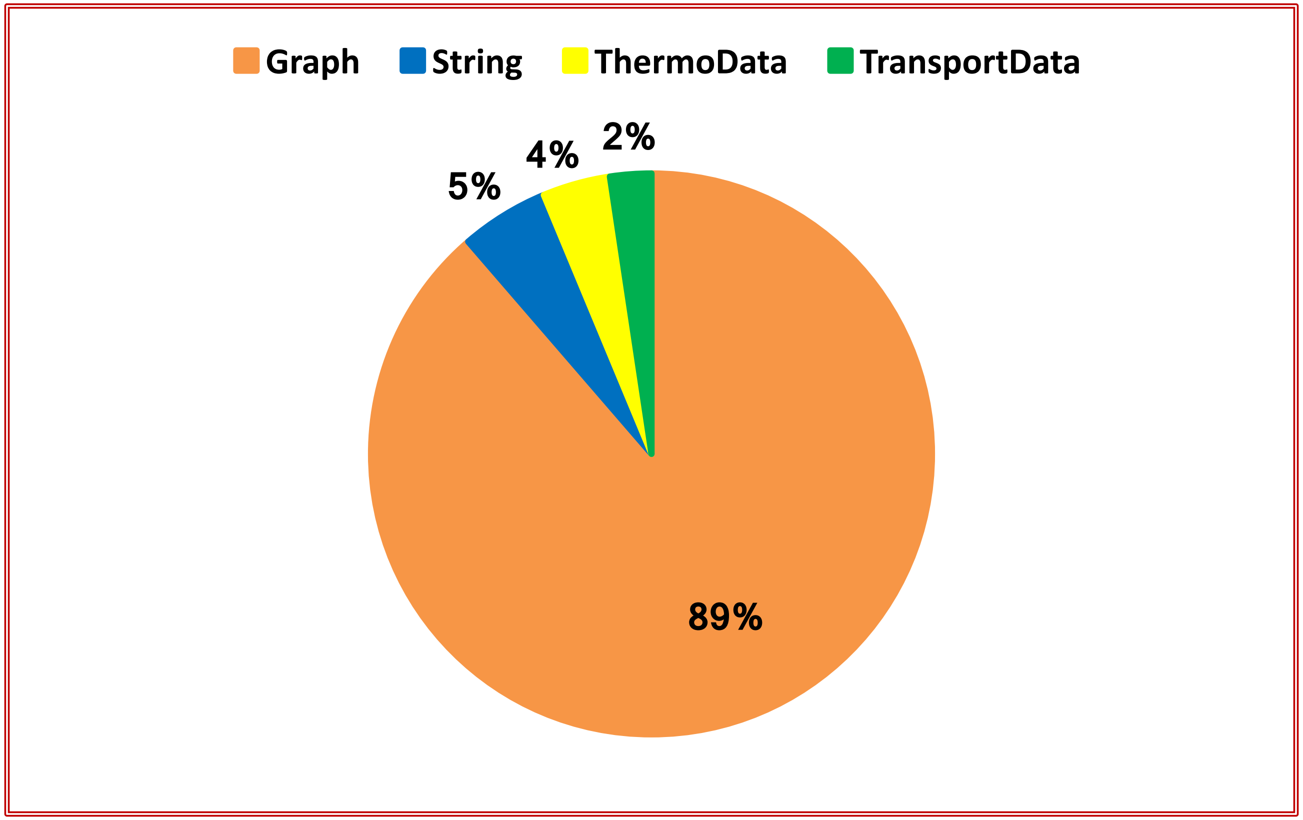
D is a new edge species? ✓

Demo of Memory Analysis



ChemGraph is the class of objects that occupy most RAM!

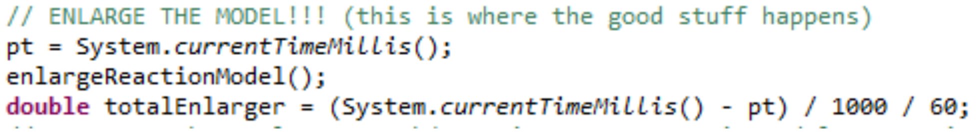
What ChemGraph Dominates?



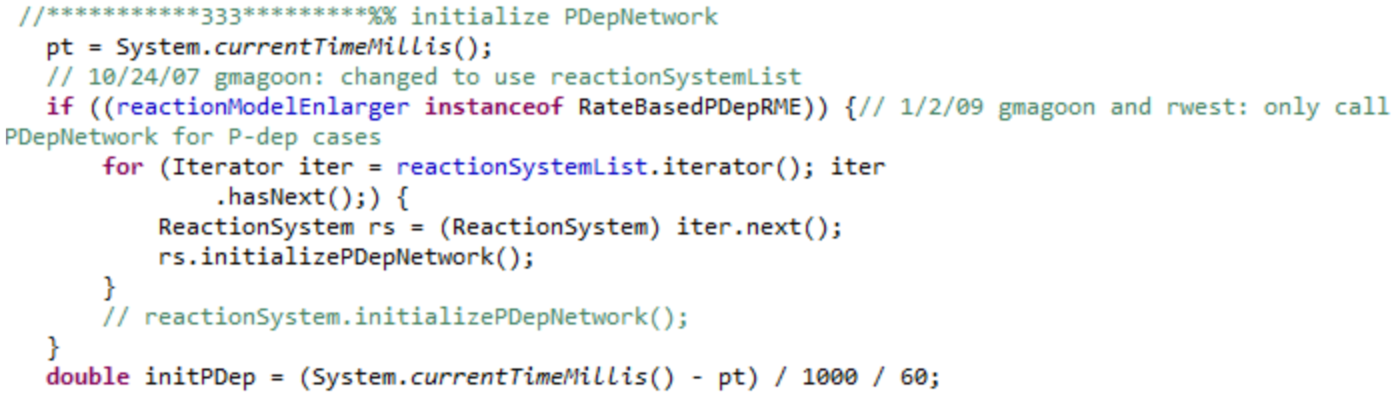
Outline

- Memory Management in Java
- Demo of Memory Analysis
- CPU-time Profiling
- Possible Approaches

CPU-time Profiling

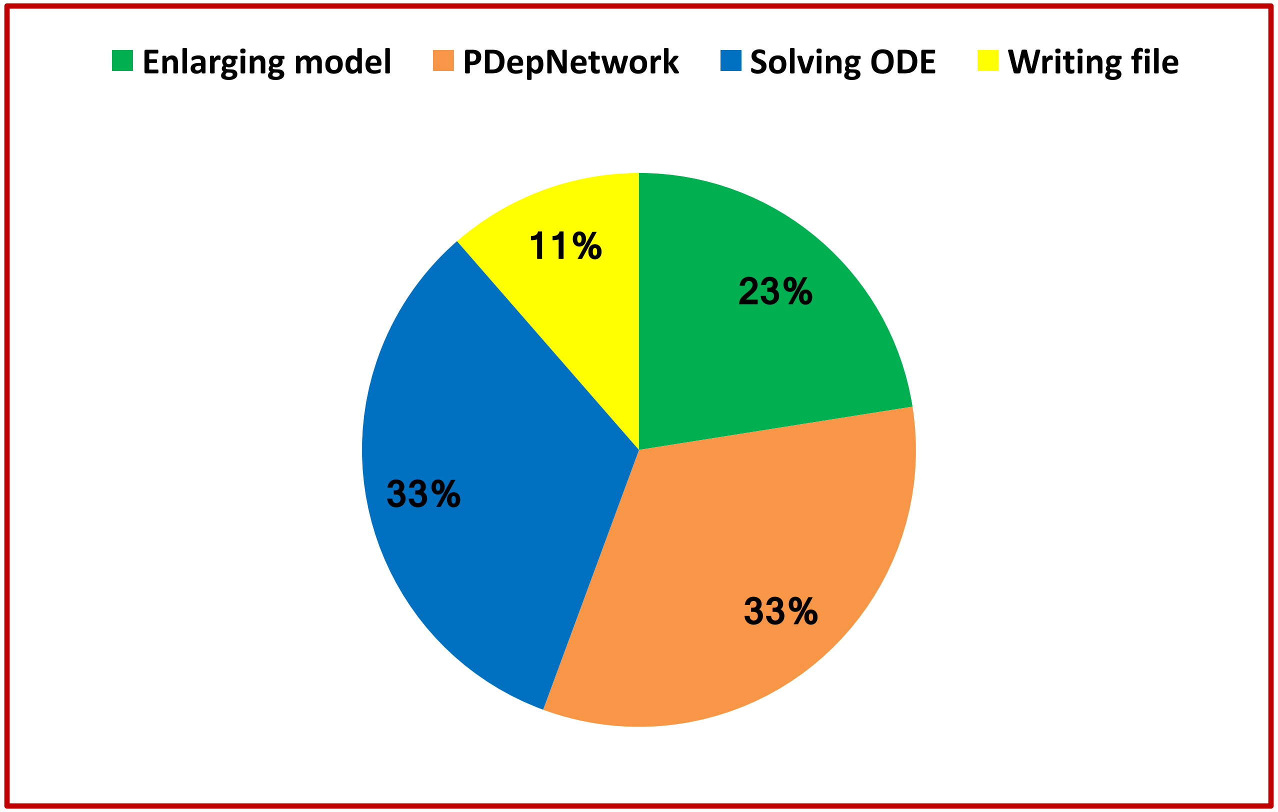


```
// ENLARGE THE MODEL!!! (this is where the good stuff happens)
pt = System.currentTimeMillis();
enlargeReactionModel();
double totalEnlarger = (System.currentTimeMillis() - pt) / 1000 / 60;
```

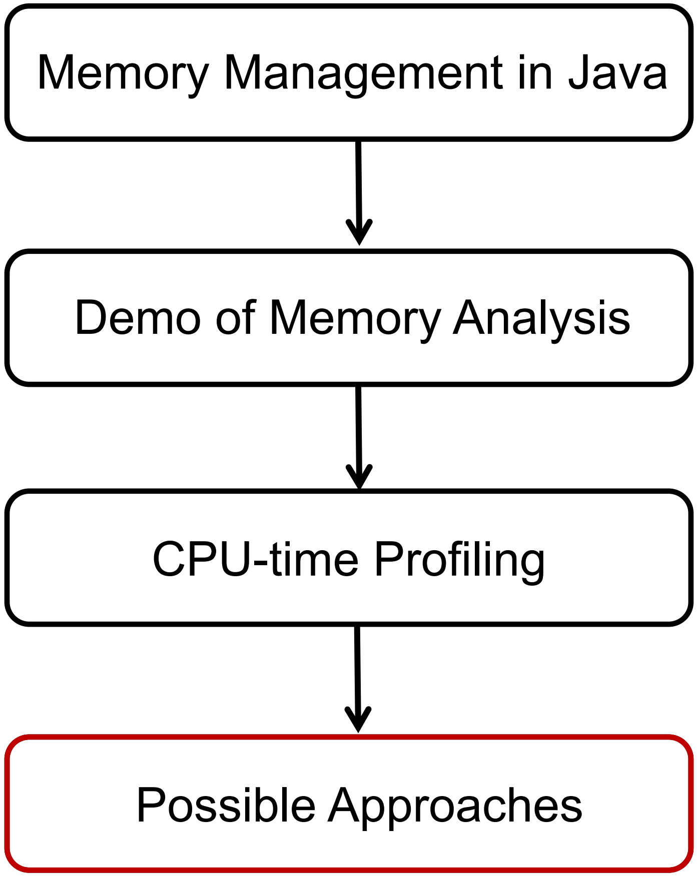


```
//**********333*********%% initialize PDepNetwork
  pt = System.currentTimeMillis();
  // 10/24/07 gmagoon: changed to use reactionSystemList
  if ((reactionModelEnlarger instanceof RateBasedPDepRME)) {// 1/2/09 gmagoon and rwest: only call
PDepNetwork for P-dep cases
      for (Iterator iter = reactionSystemList.iterator(); iter
              .hasNext();) {
          ReactionSystem rs = (ReactionSystem) iter.next();
          rs.initializePDepNetwork();
      }
      // reactionSystem.initializePDepNetwork();
  }
  double initPDep = (System.currentTimeMillis() - pt) / 1000 / 60;
```

CPU-time Profiling



Outline



Memory Management in Java

Demo of Memory Analysis

CPU-time Profiling

Possible Approaches

Approach1: Memory Usage Reduction

At later stage of reaction generation:

- ChemGraph takes up **most** memory,
- **> 95%** ChemGraphs are for edge species.

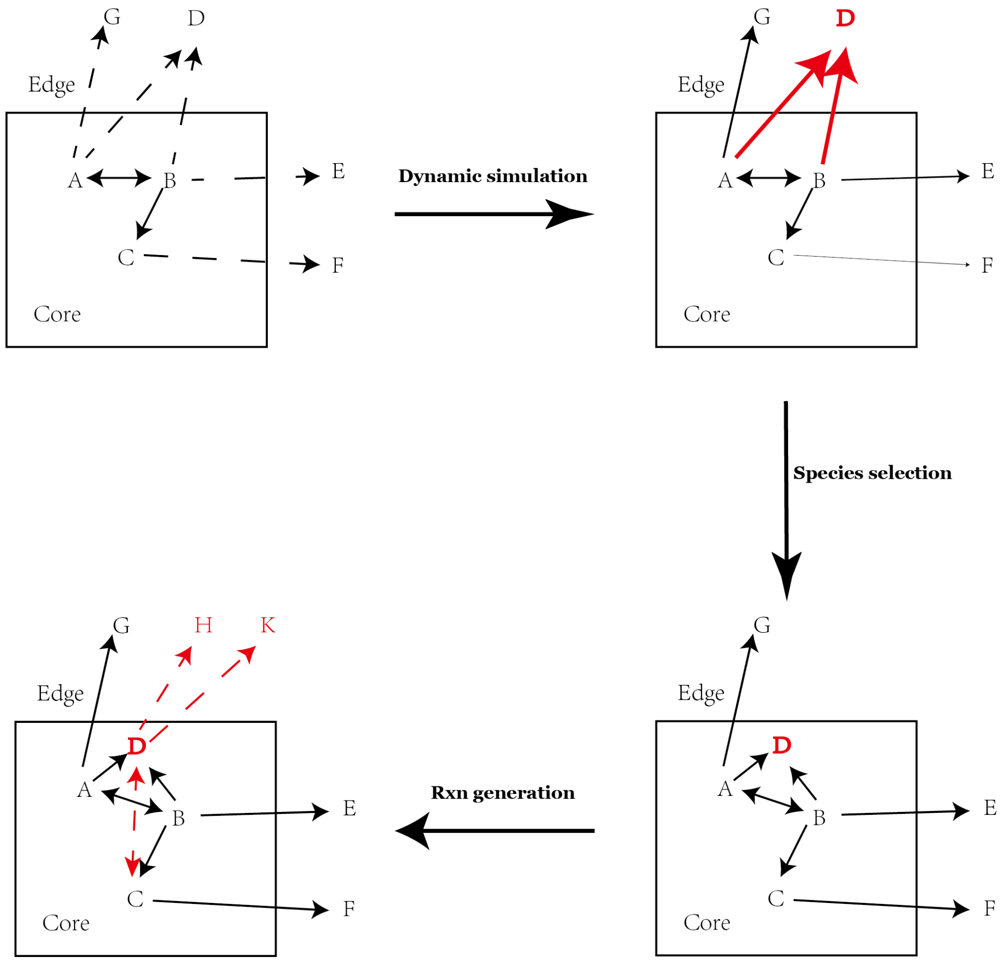
Most ChemGraphs occupy memory but contribute little

Proposed approach:

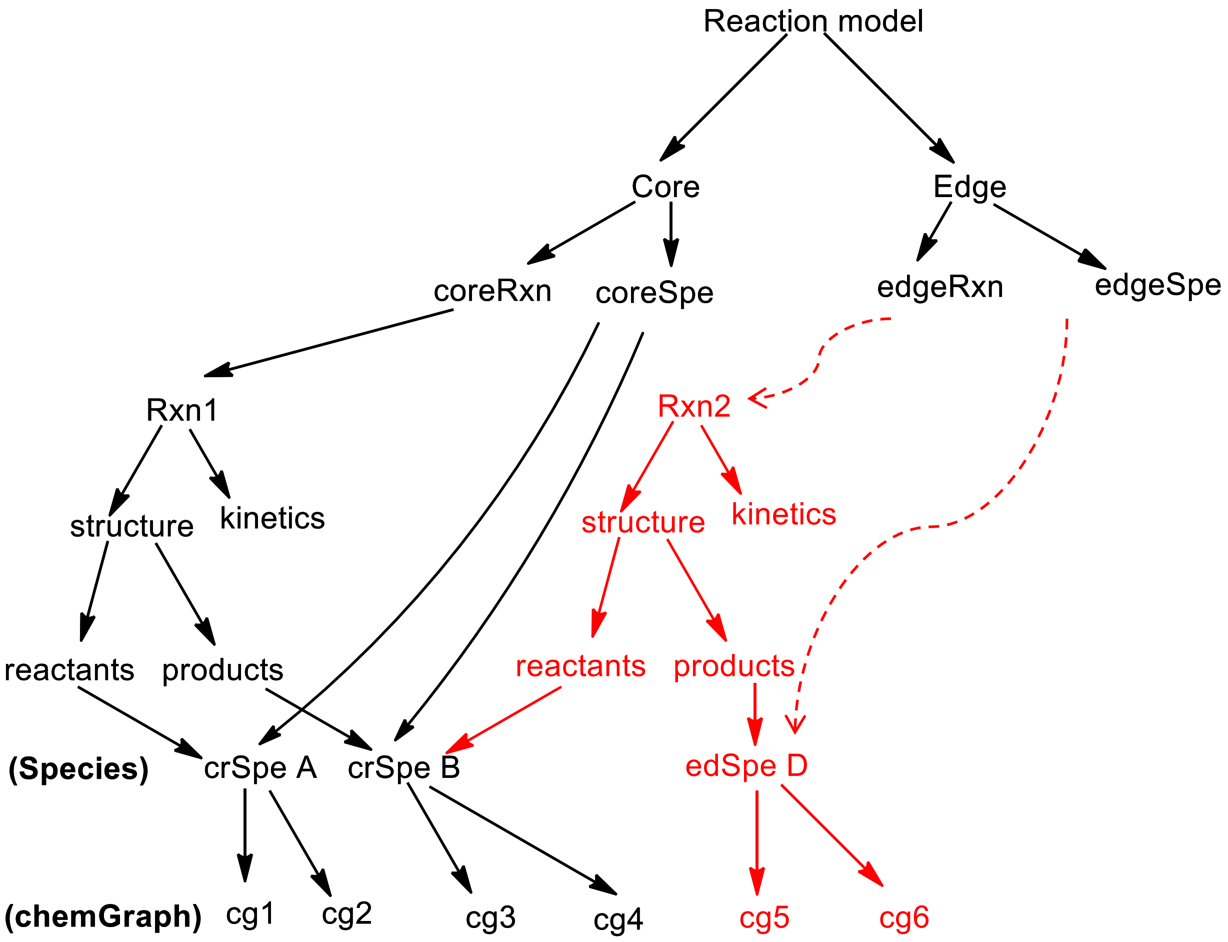
Replace edge’s ChemGraphs with much cheaper identifiers

- One identifier < 100bytes, while one ChemGraph ~ 10^4bytes,
- Can retrieve ChemGraphs back when needed,
- Can compare with other edge species using identifiers.

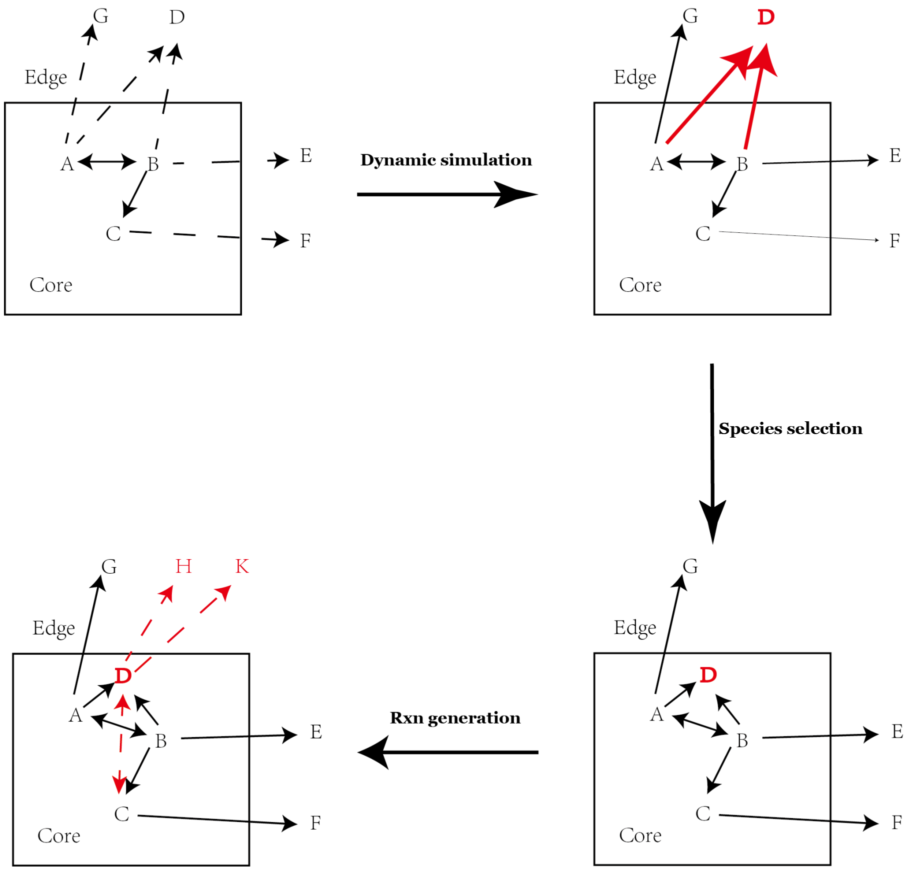
One iteration from view of ***MEMORY***



Upon Reaction Generation



New steps added

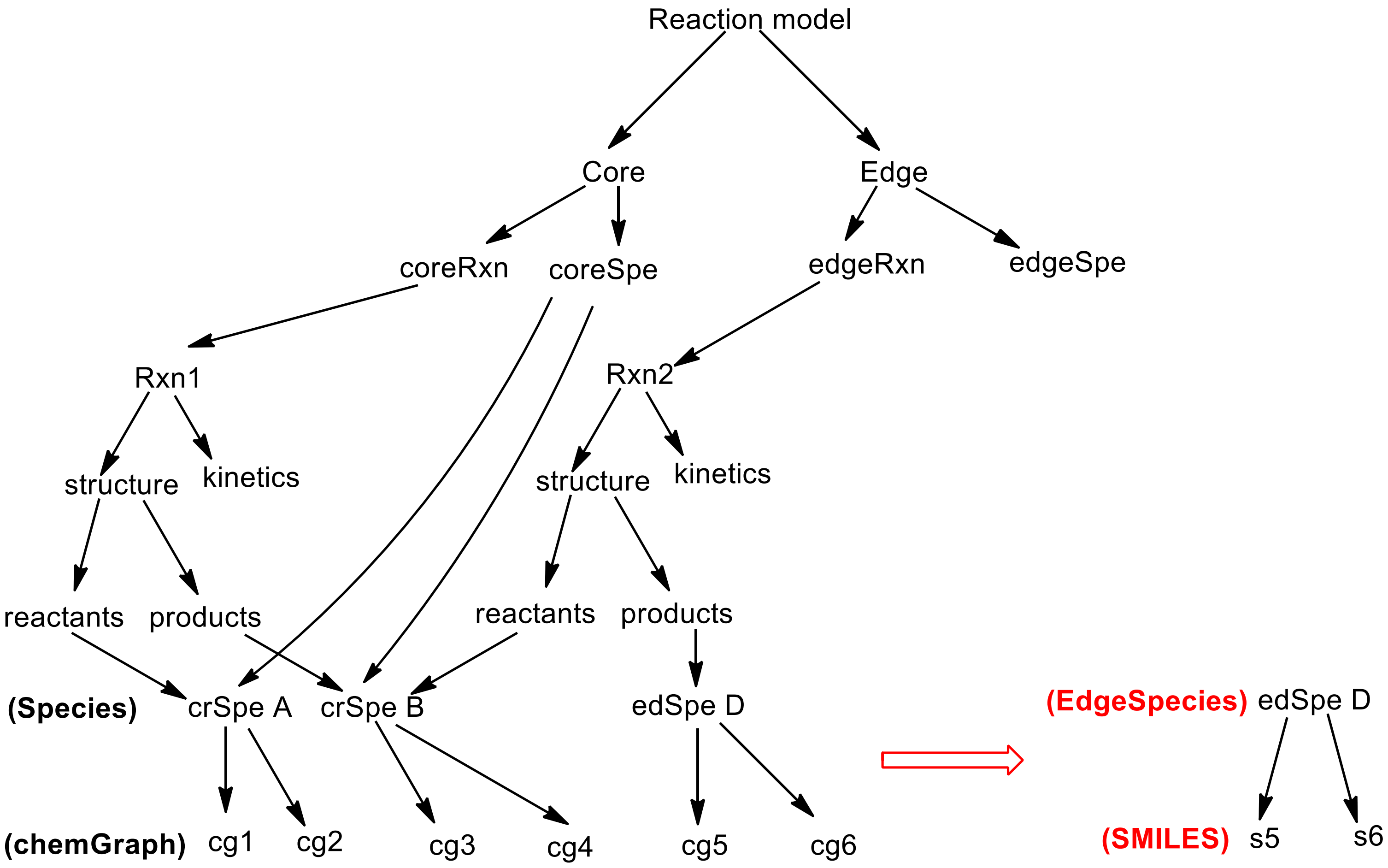


In original design, Dynamic simulation is the next step;

Now new steps added *BEFORE* that:

Memory Usage Reduction Method

ChemGraph ➔ SMILES



Reaction model

Core

Edge

coreRxn

coreSpe

edgeRxn

edgeSpe

Rxn1

Rxn2

structure

kinetics

structure

kinetics

reactants

products

reactants

products

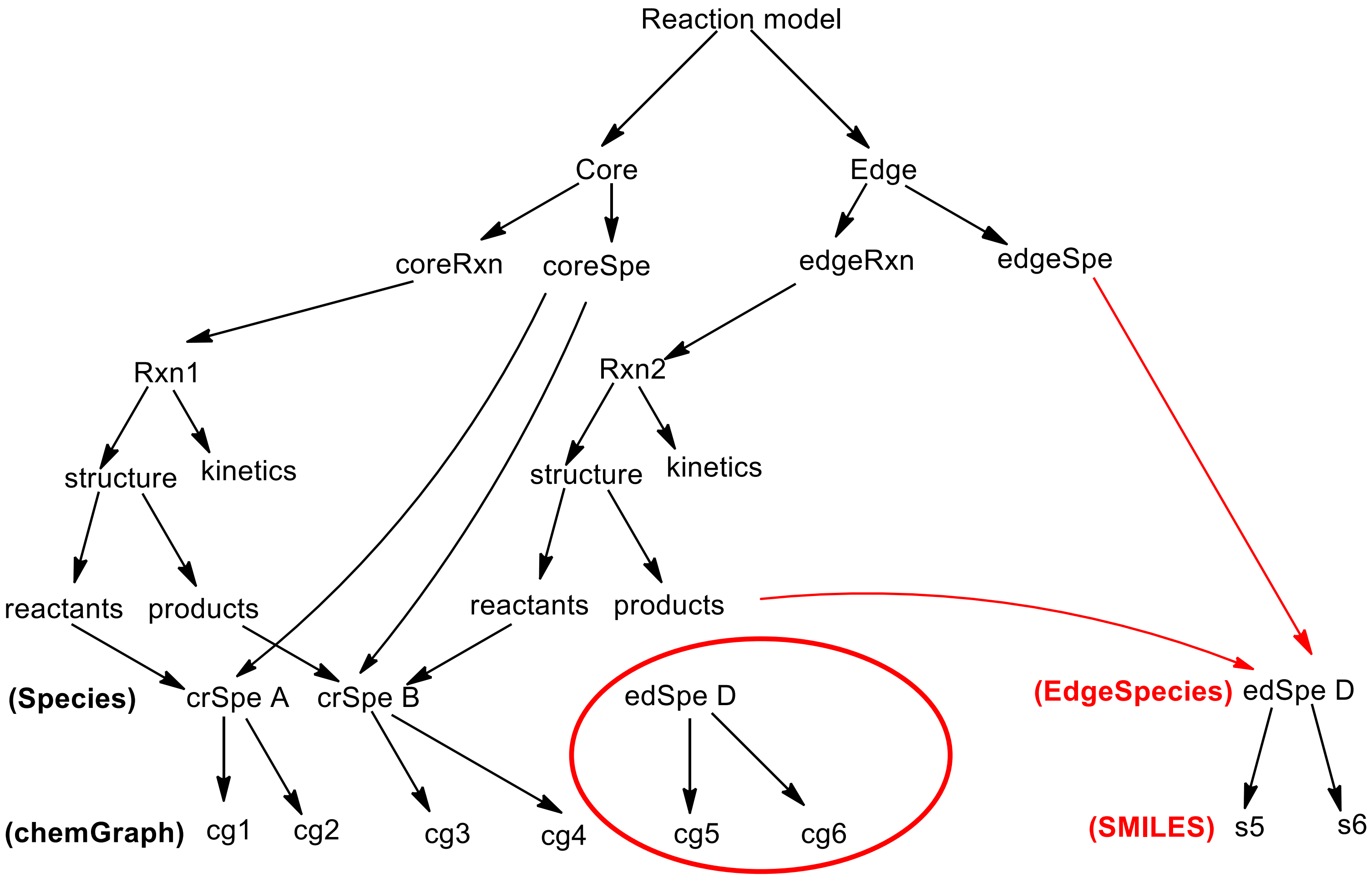
(Species) crSpe A crSpe B edSpe D

(chemGraph) cg1 cg2 cg3 cg4 cg5 cg6

(EdgeSpecies) edSpe D

(SMILES) s5 s6

If edge species D is a ***new one***


Reaction model

Core

Edge

coreRxn

coreSpe

edgeRxn

edgeSpe

Rxn1

Rxn2

structure

kinetics

structure

kinetics

reactants

products

reactants

products

(Species) crSpe A crSpe B

edSpe D

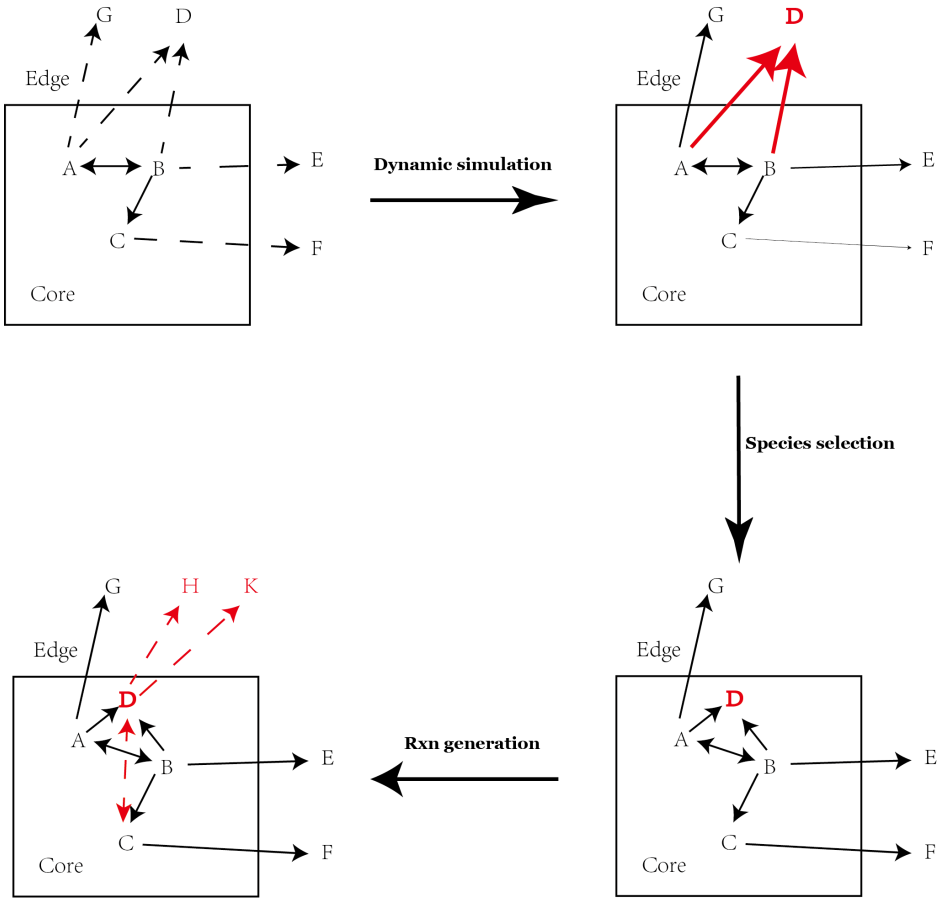
(EdgeSpecies) edSpe D

(chemGraph) cg1 cg2 cg3 cg4

cg5 cg6

(SMILES) s5 s6

Garbage collected!

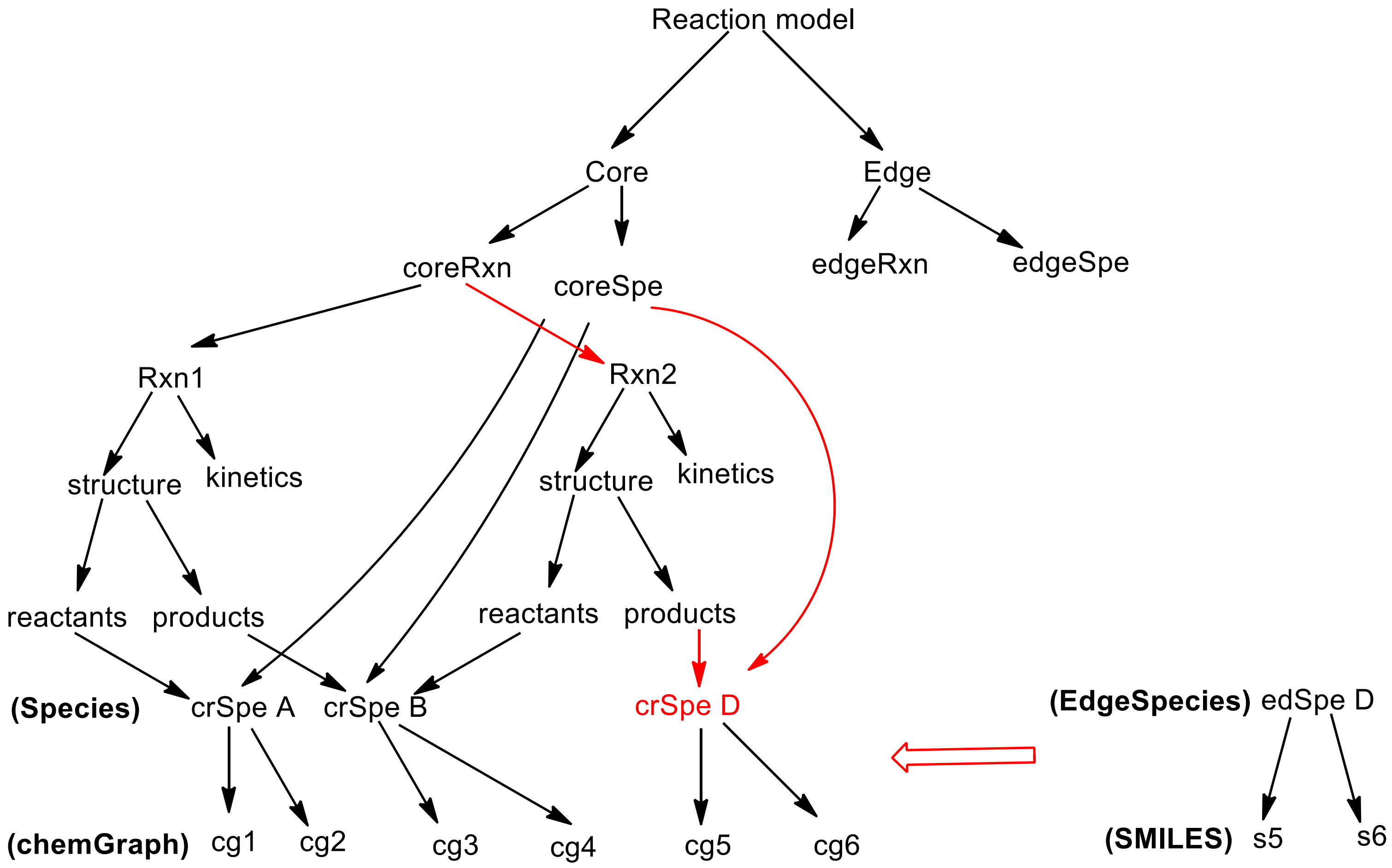


Now comes

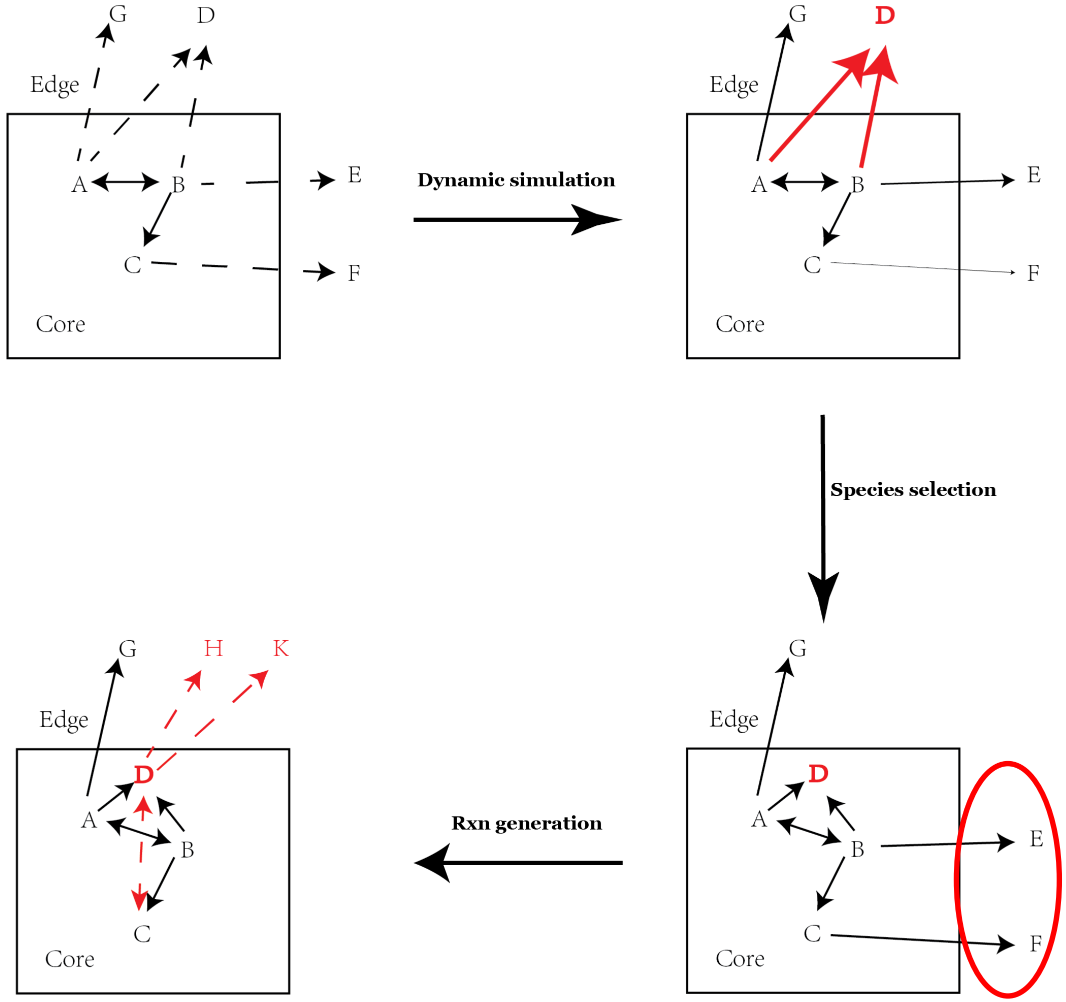
Dynamic simulation & Selection

Edge species D will be finally entering core

Upon Species D ***being selected***



<u>Approach2</u>: Pruning Edge Species

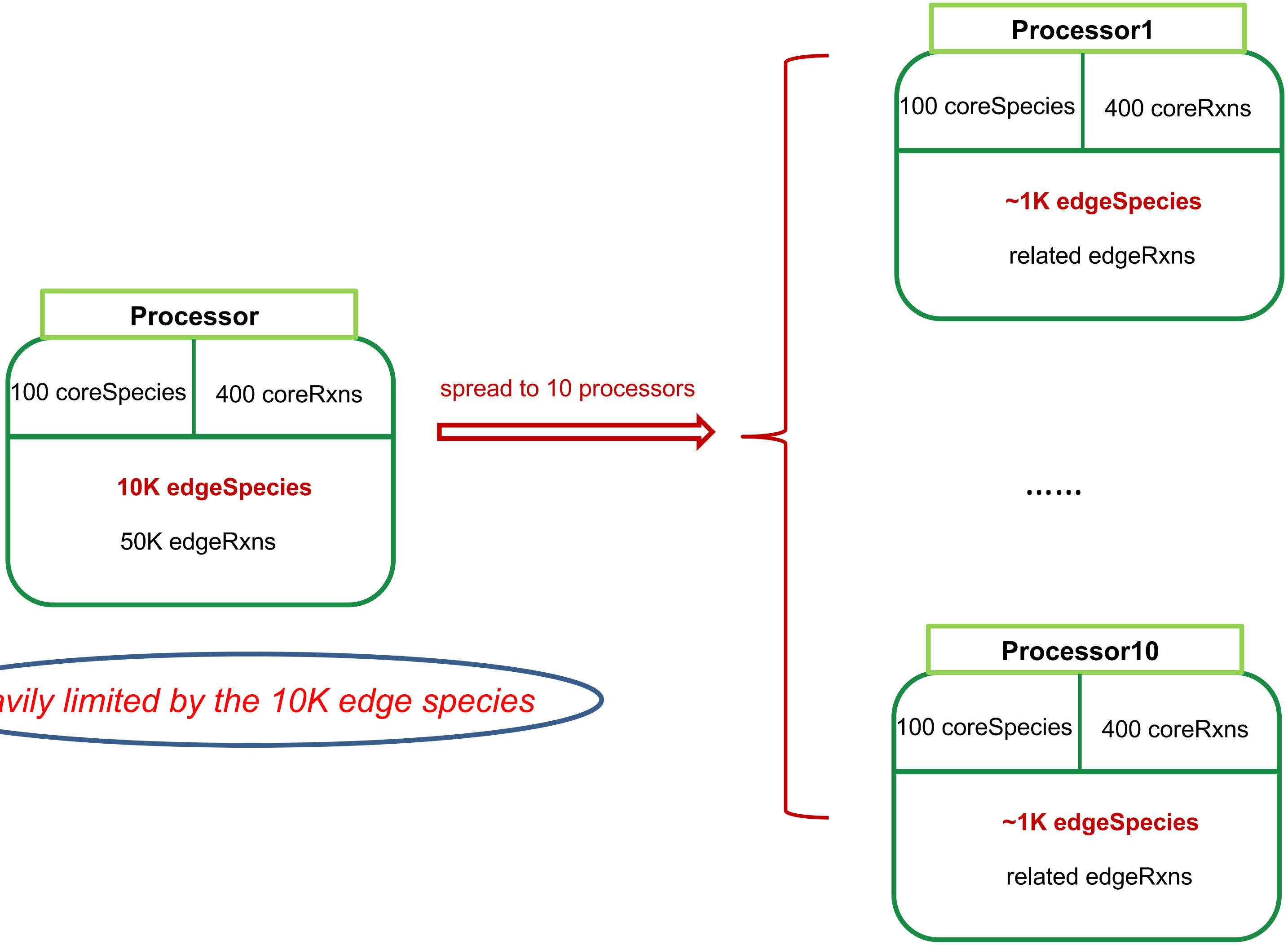


Pruning will be done based on fluxes.

- Upper limit of edge species
- Below a certain flux

Pruned!

Approach3: Job Partition



Processor

100 coreSpecies

400 coreRxns

10K edgeSpecies

50K edgeRxns

spread to 10 processors

Processor1

100 coreSpecies

400 coreRxns

~1K edgeSpecies

related edgeRxns

……

Processor10

100 coreSpecies

400 coreRxns

~1K edgeSpecies

related edgeRxns

Heavily limited by the 10K edge species

How to Partition Job

- Each processor **keeps a copy** of core model in its own memory;
- Edge species **almost evenly split** into N pieces for N processors;
  - Using M.W. makes partition easy and fast
  - Processor1 collects those species with M.W. ≤ 30
  - Processor2 collects those with $30 <$M.W. ≤ 60
  - ……
- Edge reactions **go where corresponding** edge species **go**;
  - e.g. $CH_3 + C_2H_6 \rightarrow CH_4$ (M.W.=16) + C_2H_5 (M.W.=29) should go to Processor1
  - e.g. $CH_3 + C_2H_5OH \rightarrow CH_4$ (M.W.=16) + C_2H_5O (M.W.=45)

    Processor1 stores $CH_3 + C_2H_5OH \rightarrow CH_4$ (M.W.=16) + "other edgeSpecies"

    Processor2 stores $CH_3 + C_2H_5OH \rightarrow C_2H_5O$ (M.W.=45) + "other edgeSpecies"

How Job Runs Differently

Step1: ODE solving. (Not affected)

- Edge species **don’t serve as reactants**
- Core species and edge species are **decoupled** in ODE system
- ODE solver in each processor stops **at different conversion**

Step2: select new core species. (Need communication)

- Processor with **smallest conversion**

Step3: update core and edge model.

- Move the new core species from edge to core
- Move related edge reactions to core except those having “other edgeSpecies”
- Make reactions between new core species and old core species
  - ○ Not all products are core species → checking where to go
  - ○ All products are core species → checking reverse reactions

How Job Runs Differently